Securing RDS Broadcast Messages for Smart Grid Applications

Monageng Kgwadi and Thomas Kunz

Technical Report SCE-09-06

Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada

April 2009

Abstract

Efforts to reduce peak electrical demand has led to the introduction of demand response programs for residences. Demand response programs allow customers to reduce or shift their electrical consumption from peak periods in response to dynamic prices of electricity. Utility companies broadcasts the prices to the customers who then respond by reducing consumption during peak periods or shift the consumption to off-peak periods. Similarly, direct load control programs entice consumers with special rates or other incentives for allowing the utility to control load (typically air conditioning) for a number of days per year. Both uses require a ubiqituous and cost-effective communication network to allow utilities to communicate with users and appliances. The Radio Data System (RDS) has been identified as one strong candidate technology. However, security concerns arise due to the wireless nature of the communication channel. Source authentication is crucial in demand response to ensure that only authenticated messages are responded to.

This report presents evaluations of cryptographic methods that could be employed to offer source authentication over the RDS network. Simulations are used to determine the impact on the network performance by employing digital signatures to allow source authentication. The simulations were calibrated with data collected in Ottawa, Canada, in particular to model signal propagation characteristics. While different environments experience different path losses, the relative comparisons are not impacted by this difference, however. The authentication schemes studied all provide strong authentication against attackers who attempt to forge signatures without knowledge of private keys (which are held at the transmitter). The information exposed in the transmitted messages will not help an attacker in forging future messages. And as messages are time-sensitive and the senders and receivers in the network coarsely time-synchronized, replay-attacks are prevented as well. This is different from shared-key/secret-key schemes such as the one employed in Zigbee, where exposure of the secret key on the receiver side (using bit sniffing or other techniques to access the non-volatile memory on the receiver) compromises the security/authentication of messages.

The report presents comparisons of the security offered by the protocols, the bandwidth overhead, computational costs and message reception probability. Our simulation results show that, up to a distance of 90 km, all authentication schemes do not affect message reception by the receivers. Beyond that, all the schemes have an effect on message reception due to increased message sizes and receiver bootstrapping for BiBa and HORSE. ECDSA and HORSE outperform BiBa in terms of message reception beyond 90 km and the difference between the two is not significant. ECDSA however offers higher security than HORSE and BiBa but at the cost of increased computational complexity, in particular at the receivers. In addition, ECDSA has the highest bandwidth overhead.

Contents

1 Introduction

The need to reduce peak electrical demand has resulted in plans to introduce demand response programs for residences. Demand response programs allow customers to adjust their electrical consumption in response to dynamic prices of electricity. In such programs, the utility company informs the customers of a price event to which the customers choose to respond. Price events alert customers of a change in the prices of electricity. Customers would then minimize their consumption during periods with high prices, resulting in lower power bills. Emergency events deal with issues concerning grid reliabilities. In the event of a grid instability, the utility company issues emergency events which compel customers to lower their consumption, thus reducing the load and alleviating the strained grid. The use of demand response provides utility companies with another option to perform load management in addition to load shedding and power purchase. Demand response uses both the price events and emergency events to improve reliability and lower electricity costs to customers.

The Programmable Communicating Thermostat (PCT) system is aimed at reducing electrical power consumption during peak demand periods. The PCT system allows demand response programs to be applied on the power consumed by air conditioning of residences. The system allows thermostats to receive pricing events broadcast over a wireless communication channel from the utility companies. The programmable thermostats operate in an automated manner. A customer programs the thermostat and the thermostat responds to event messages accordingly. The operation of the thermostats, and consequently demand response programs over the PCT system, rely on successful delivery of valid event messages. Therefore, there is a need to ensure the integrity of the messages and authenticate their origin.

In addition to the above use of messages to enable users to respond to dynamic energy prices in residences, direct load control that does not use dynamic energy prices. Instead, consumers are provided special rates or other incentives for allowing the utility to control load (typically air conditioning) for a number of days per year. Also, the original PCT system concept has since been expanded to include a range of Programmable Communicating Devices (PCDs), including PCTs, in-home displays, smart appliances and control switches for air conditioning, water heaters and other high energy consuming devices. In future extensions, plug-in hybrid vehicles could also be included. However, in all these extensions of the original idea, the overall concept is the same: utilities send messages to devices to inform users and to potentially directly control the load. Therefore, the same need for reliable message delivery, message integrity, and proper authentication arises. In the remainder of this report, we will use the PCT system as the example PCD system, as this work was started with respect to the requirements identified for the PCT system. But the key insights and solutions apply equally well to a more general definition of a receiver device, be it a thermostat, in-home displays, or smart appliances.

The draft reference design for PCTs mandates a nation-wide wireless broadcast communications network using either the Radio Broadcast Data Network (RBDS or RDS) or the Paging system [1]. Later revisions identified the RDS as the communications infrastructure to be employed for the PCT system. The wireless nature of the communication infrastructure puts the smart grid applications running over it at a security risk. The

RDS network does not employ any security mechanisms on which the PCT or a PCD system can rely. Therefore there is need to provide for secure means of communicating PCD system messages over the RDS network. This report proposes solutions to address the security over the RDS network. The solutions presented in this report can be employed to authenticate messages sent by any application using the RDS network as the physical infrastructure. A PCD or PCT system is an immediate beneficiary to the service, hence it is presented as a test case. In [2], the possible security threats to the PCT system are studied and a risk management approach is used to propose mitigation steps for the security concerns. This report provides an analysis of the security threats for a communication protocol for use with PCTs. A literature survey of security issues in similar networks is carried out to identify solutions that could be used or extended to the PCT system. Of particular interest are sensor networks and RFID networks because they face similar challenges of limited resources. The report also identifies possible solutions that could be pursued to provide authentication over the RDS network and their impact on the network. Three authentication schemes identified to be suitable for the RDS network,(BiBa, HORSE, ECDSA) are investigated using simulations to determine the impact on network resources.

A background on the security of the PCT system, RDS network and security implications is given in Section 2. The threat model as relevant to the PCT system employing the RDS network for communication is presented in Section 3. A literature survey of security schemes available is presented in Section 4. Section 5 gives detailed description of three authentication schemes suitable for the RDS network and how they can be employed to authenticate messages. Analysis of simulation results of the three authentication protocols is then presented in Section 6. A conclusion is then presented in section 7.

2 Background

An initial study of the security characteristics of the PCT system in [2] advocates a tiered security solution. The solution defines the System Owner as responsible for overseeing and controlling the PCT system. All the messages that the System Owner sends to the PCTs go through the System Operator. The System Operator is responsible for delivering the messages to the PCTs within its geographical or logical coverage area. The goal of our study is to provide secure communication between the System Operator and the PCTs using the Radio Broadcast Data System (RBDS or RDS) network.

The Radio Data System (RDS) was designed to carry small packets on the FM channel in the range of 87.5MHz to 108.0MHz. It has been used to convey program information and traffic information to radios in vehicles [3]. The Radio Broadcast Data System (RBDS) is the North American equivalent of the Radio Data System (RDS) which is a European standard. In this document the terms RBDS and RDS are used interchangeably. RDS is a broadcast one-way communication channel and has no security features that the PCT system can rely upon. This leaves it to the application using the RDS network to do the required security and authentication. This means that messages that require security need to be encrypted to ensure security. In the PCT system, privacy is not as much a priority as authentication. The event messages are to be broadcast to alert everybody about events, therefore there is no need to make such messages secret. Authentication however is necessary to ensure that only authenticated messages are responded to. As pointed out in [2], an attacker could cancel events prior to their intended period elapses. The attacker in this and many other ways can cause distress and possibly cause grid instabilities, effectively defeating the whole purpose of demand response. Therefore, the PCTs have to authenticate the origin of the message and only respond to an authenticated sender(s).

There is need to provide for security in the design of the communication protocol as recommended by [2]. The security of such a system should be resilient to attacks and be able to recover easily from a breach. Bono *et al* show in [4] that obscurity is not a good measure for ensuring security. They advocate the use of standard cryptographic algorithms employing keys of sufficient lengths. They demonstrate this by bypassing the immobilizer of a vehicle which employs a cryptographically-enabled RFID tag. They achieved this by reverse engineering, key-cracking and simulation. The immobilizer in their study employed a Texas Instrument Digital Signature Transponder (DST). From the knowledge of a rough schematic posted on the Internet, they were able to determine the functional details of the cipher of the DST. The challenge/response authentication messages between the reader and the tag were obtained and used to crack the key. The 40-bit shared secret key was extracted with the use of an array of FPGAs in less than an hour. Then, using the extracted key, they were able to simulate the RF output to spoof the reader. In their study, they were able to establish conditions for hot-wiring a car with fairly modest resources.

Strong cryptographic algorithms add to the complexity and ultimately the cost of manufacturing the devices. The price of the PCTs has to be minimized as they are expected to retail at less than $50 [2]. Sensor networks and RFID networks face similar problems with the need to provide security and still keeping the cost of the devices relatively low.

It could be expected that the PCTs may have slightly more computing resources, storage, and power supply than RFID tags and sensor nodes. However, the PCTs are still expected to have modest computing and storage resources compared to today's computers. This limitation means that the security and authentication algorithms employed on these devices be efficient and low cost.

Authentication poses more of a challenge in a one-way communication channel because conventional authentication methods of challenge/response cannot be used. In a challenge/response authentication, a sender proves its identity to the receiver by responding to a challenge from the receiver and vice versa. This cannot be done over the RBDS network, since the PCTs do not have a communication channel to the System Operator with which they can challenge the identity. Even if such a channel existed, the volume of challenges coming from the PCTs would be too high to make this approach attractive.

3 Threat Model

The characteristics of an adversary and the impact of the threat posed need to be established before discussing mitigation strategies. The PCT system is subject to a number of attacks as stated in [2]. The adversary that we discuss in our study is limited to one who attacks the PCT system via the wireless communication channel. The motives of such attackers could be anything from leisurely mischief to a terror attack targeting denial of utility services to customers. According to [2], the attacks that an adversary could launch on the PCT system include, but is not limited to, the following :

- An attacker could cause unanticipated loads on the grid causing instabilities by sending false messages to customers. This could be done by canceling valid emergency event messages aimed at alleviating existing grid instabilities thus preventing the expected reduction in load.
- An attacker could send false time synchronization messages creating erroneous behavior of the PCTs.
- Customers could be deceived by false messages displayed by the PCTs if an attacker can successfully send such messages to the PCTs.
- A successful breach of the communication can allow an attacker to shut down PCTs or even install new software into the devices. An attacker who is able to shut down PCTs remotely could cause irritation, discomfort and health problem to some users. The installation of new software (potentially malicious) by an attacker may lead to erroneous operation of the PCTs.
- An attacker could jam the signal to a subset of receivers from a ground station or aircraft e.g. balloon.

A systematic risk analysis of the threat posed to the PCT system and counter measures were fully described in [2]. For the purpose of this report we address the threat and mitigation procedure for a PCT system employing the RDS network to communicate messages.

The nature of the RDS network limits the way an attacker can launch attacks. The lack of a reverse communication channel from the PCTs to the System Operator means that the attacker should have physical access to a PCT to access information on it. We assume that an attacker has unlimited access to PCTs, either from his/her own home or he/she could break into someones home and access a networked PCT. Moreover, an attacker could easily purchase the device at a retail store. This means that the data stored on these devices can be retrieved by a determined attacker using any method at his/her disposal. This setting does not bode well for security by obscurity of cryptographic keys. We assume it would be fairly easy for an attacker to retrieve a decryption key(s) from the PCT, thus the use of symmetric key cryptography should be avoided. Asymmetric cryptographic methods are more favorable for this setting. If public-key cryptography is used, an attacker would only retrieve public keys of the System Operator by attacking the PCTs. An attacker would be forced to attack the Systems Operator to obtain the private keys that would allow him/her to encrypt messages.

Although the attacks on the PCTs are easy, as mentioned above, attacks on the sender (System Operator) are not trivial. An attacker wishing to get information from the sender is limited to eavesdropping or gaining direct (or indirect) access to the system information database. The former method of attack means that the attacker is limited to what is communicated and what is stored on the PCTs. Methods of communication that reveal no information to an eavesdropper and store no critical information at the receiver should be employed to lower the risks of this type of attack. Gaining physical access to the system database is not easy but none-the-less possible. An adversary could break into the premises and obtain critical cryptographic information that would allow him/her to launch an attack. The critical information used for communication could also be leaked out through employees to an adversary by negligence, blackmail, extortion or ignorance, to mention a few. A decentralized method of storing cryptographical information should be used to avoid a single point of compromise. To protect the cryptographical information, [2] suggests that complementary pieces of cryptographic information should not be stored in one place or exposed to one person.

Operation on the FM radio spectrum requires licensing from the radio spectrum management organization. An attacker operating unlawfully on a frequency without a license would be stopped if detected. As part of their non-cryptographic solution to provide security, [2] proposes the use of monitors placed to detect infringements. The monitoring devices should be conveniently placed to receive the messages and compare them with those sent by the System Operator. These devices, carefully placed in the coverage area, will reflect what the PCTs receive from the network. With such measures in place, it would be easy to detect if the messages are tempered with or if there are new unaccounted messages showing up at the PCTs. The authorities then would be alerted of the infringement. The use of a detection system cannot be relied upon to provide security. The attacks on the PCTs can go unnoticed if they are targeted to a small subset of customers who are in the blindspot of the monitors. Moreover, the attacker could be mobile and operate for a short period and leave no trace. In such cases it would be very difficult for authorities to stop future attacks by the same attacker even if such attacks are detected.

4 Literature Survey on Possible Mitigation Steps

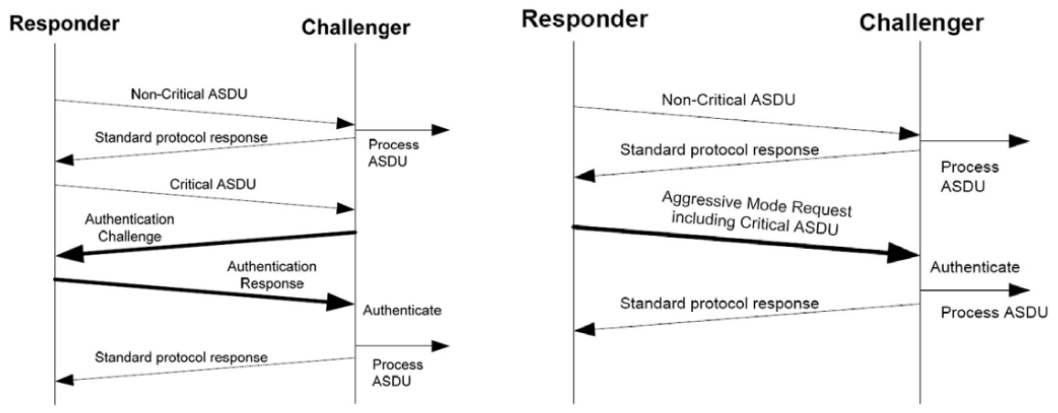
Several symmetric and asymmetric cryptographic methods exist in the literature. A survey that covers the technical problems faced by RFID security and privacy is presented in [5]. Several cryptographic methods are proposed in the literature for RFID tags in [6] and [7]. The methods of cryptography favored by the security experts consulted in [2] is the use of Elliptic Curve Cryptography (ECC). The following methods have been identified as promising to the application for electrical demand response in residential devices. The cryptographical authentication methods presented here were designed for networks different to the RBDS network. However, these could be extended or customized to use on the RBDS network to offer the required security.

4.1 Secure SCADA using DNP3

Related developments in the power industry have been carried out to provide secure communications for metering and Supervisory Control and Data Acquisition (SCADA) services over the existing power-line infrastructure [8]. The Real-Time Energy Management via Power-line and Internet (REMPLI) system was designed to permit remote and autonomous control and monitoring of energy resources by utility companies at residences [8]. The design requirements for the system were to allow various utility companies (e.g. electricity and gas) to offer their services to customers over a shared distribution network. The services required for support by REMPLI include load balancing, theft detection, remote monitoring and control.

Mander *et al* in [9] discuss a distributed security architecture using the Distributed Network Protocol (DNP3) to offer security for residential load-management devices. Their solution protects Intelligent Electronic Devices (IEDs) networked to a SCADA network from cyber attacks. The DNP3 protocol is used widely in the world for electricity and water utilities for communications with field equipment [10]. DNP3 does not provide sufficient security features, hence their solution extends DNP3 by adding a security layer and using data object security [9]. The security extension to DNP3 provides authentication at the data-link layer by using encryption. The security provides privacy to customers and conceals the events that IEDs are experiencing, which an attacker can exploit to attack and disrupt the system. Moreover, encryption prevent burglars from determining, based on the energy consumption, if a house is occupied.

Figures 1(a) and 1(b) show how NDP3 is used for authenticating application messages between two entities. DNP3 employs the challenge/response authentication method to authenticate critical commands as shown in Figure 1(a). The receiver queries the identity of the sender upon receiving a critical command. The sender proves its identity by demonstration of knowledge of shared cryptographic keys. The aggressive mode as shown in Figure 1(b) is used to conserve bandwidth by eliminating the challenge and response messages. In the aggressive mode, the authentication data is included in the message. The aggressive mode is considered slightly less secure than the normal challenge/response mode [10]. For the aggressive mode to be used there has to be at least one request/response authentication preceding it to establish trust between communicating entities.



(a) DNP3 challenge response mode (b) DNP3 aggressive mode

Figure 1: Modes of DNP3

The security extension to DNP3 in [9] uses two levels of security. The first level of security is achieved by using symmetric operations, while the second level employs asymmetric encryption operations. The asymmetric operations are used to exchange the keys used for the symmetric operations. All the other data is encrypted by the symmetric operation and carry no security layer header. This is efficient for bandwidth constrained links by minimizing the overhead [9]. Each device uses different symmetric keys for transmission and reception of messages. This ensures that if an attacker is able to crack the key in one direction, they cannot access the data in the reverse direction. The security layer updates the symmetric keys in an asynchronous manner i.e. there are no predefined times when the keys are refreshed. When a new key is used, the receiving IED will try to decrypt using the old key which results in a decryption error. When a decryption error results from a decryption operation, the IED tries the next valid key. If the decryption is error free, the new key is used as the current key. If however the decryption yields another error, the old key is reinstated as the current key.

The key exchange for the asymmetric operation in a one-way communication network such as the RDS network is a problem. The Diffie-Hellman algorithm as suggested in [9] cannot be completed. A different approach is necessary for the exchange of the public keys during initialization or in the event a private key is compromised. The authentication of the symmetric operation relies on the asymmetric key exchange. As it is described in [9], an adversary needs only the senders public key to extract the secret symmetric key during key exchanges. With the symmetric key, an attacker is able to launch attacks. The symmetric operations would fail to ensure authentication if the key and cypher are known to the attacker. The attacker can encrypt messages using the secret key and the receiver would perceive the attacker as authenticated based on the demonstration of knowledge of the secret key.

To address the above shortcomings in the context of RDS, the symmetric operation could be replaced with an asymmetric operation. Public key cryptography could be used in place of the AES symmetric operation to minimize the threat if the attacker is able to obtain the key from key exchanges. If public key cryptography is used in place of the symmetric operation, the key exchange would involve only the public key. Even if the attacker is able to obtain the public key, he/she cannot authenticate him/herself to the receiving device without knowledge of the private key. The key distribution issue still persists with this approach. The initial set up of the session keys with which the message keys are encrypted needs to be made seamless and easy to update in the case of a compromised private key.

4.2 Authentication using RF fingerprints

The physical layer RF fingerprints can be used together with higher layer protocol methods to provide authentication [11]. RF fingerprints identify an RF transmitter from the properties of the received radio signals. They allow different transmitters to be distinguishable from one another. The authentication presented in [11] is for single-hop wireless sensor networks. The application forms a network with authenticated hardware instead of the users authenticating themselves. The nodes in the network have fingerprinting capabilities and are able to distinguish RF sources without knowing their fingerprints before initialization. On initialization, the nodes perform a neighbor discovery protocol and form a secure group of fully connected neighbors. The group members exchange the fingerprint information and then build credentials for group members. The credentials of group members are made up of RF fingerprints, RF identities and cryptographic identities. In this way the network nodes authenticate their neighbors.

In the RDS network, a distributed approach as described in [11] is infeasible. However, the devices could be enabled to do RF fingerprinting and use location in the credentials of the sender (System Operator). In this way an attacker masquerading as the System Operator would be forced to operate very close to the legitimate System Operator. If an attacker is forced to operate near the legitimate System Operator, then his/her effective radiated power would have to be comparable to the System Operator for his/her signal to be detectable. The total effective radiated power for systems operating in the RDS network are in the order of tens of kilowatts. The cost of equipment and operation should serve as a deterrent for most attackers. Even if the attacker was able to obtain the equipment and broadcast messages, the spectrum management regulation body monitoring the use of the radio spectrum could be relied upon stop the unlawful operation.

The use of RF fingerprints alone does not provide absolute security and has to be used with other cryptographic methods [11]. In the implementation described in [11], encryption keys and RF credential should be used to provide security. For authentication on the RDS network, the RF fingerprints are sufficient to identify the authentic sender from attackers. The PCTs should be able to learn the new fingerprints of the System Operator if the transmission RF equipment changes for any reason. The PCTs should be able to distinguish an attacker from a legitimate System Operator with changed fingerprints. Additional hardware and digital signal processing (DSP) units would have to be incorporated into the PCTs to enable RF fingerprinting. The extra hardware could potentially drive the cost of the PCTs high. The cost of such additional hardware is unknown to the author and requires investigation to determine if the solution is cost effective.

4.3 Zero-knowledge device authentication

A method that allows pre-authenticated response between RFID tags and readers was presented in [12]. The solution curbs divulging critical information by RFID tags to any random tags upon interrogation. Engberg *et al* in [12] propose a solution where tags only respond to authenticated readers . The method was developed to avoid customer tracking using RFID tags that the customer may have on them. The method employs the use of a 'zero-knowledge' device authentication method. The method is not technically conventional zero-knowledge, the authors claim zero-knowledge because the tags do not contain any sensitive data. The tags relay the requests to the user/customer upon authenticating a reader to which the customer responds.

The solution differs from the extension to DNP3 in that it does not use a challenge/response method to authenticate a sender. The solution involves a user sending a combination of a non-encrypted nonce, and a second nonce using XOR and hash functions. The receiver authenticates the sender on the grounds of knowledge of the shared secret. A zero-knowledge authentication request (ZAR) is given by:

ZAR: [DT; (RSK **XOR** **Hash**(DT **XOR** SSDK)) ; **Hash**(RSK **XOR** SSDK)]

where DT is the first nonce, RSK is the random session key (second nonce) used to encrypt messages within a given session and SSDK is the shared secret. From the message, the receiver can obtain the hash value of (DT XOR SSDK) and use it to obtain the random session key by an XOR operation with the second part of the message. The third part of the message is used to verify the random session key and authenticate the sender. From the ZAR, the RFID tag can authenticate the reader and only respond to authenticated readers. The first nonce (DT) also serves to protect against replay attacks and the use of Date Timestamp is advocated in [12]. The tag can then respond if the authentication is valid. The protocol guards against fake acknowledgments by using a zero-knowledge acknowledgment that is a function of the shared secret.

The implementation described above employs a shared secret key, but could be extended to use asymmetric methods as well [12]. The zero-knowledge protocol appears to be favorable for the one-way communication channel like the RDS network. Provided the secret keys are distributed effectively, it requires only one ZAR to authenticate the sender. The absence of a challenge/response operation for authentication favors deployment in a one-way communication channel. The PCTs would be able to authenticate the System Operator based on a single ZAR. Then the session key could be used to encrypt all the messages of the session. The operation could be used in connection with that of the DNP3 discussed earlier. RDS offers a relatively low bandwidth channel and the message size of ZAR could be large depending on the sizes of the exchanged data (the two nonces). Therefore, the frequency of sending such messages should be minimized to ensure efficient use of network resources. A customized ZAR could be employed for use on the RDS system to reduce the overhead. For the purpose of the PCT system, the RSK could be excluded in the ZAR to reduce the communication overhead.

4.4 The TESLA Broadcast Authentication Protocol

The Time Efficient Stream Loss-tolerant Authentication (TESLA) broadcast authentication protocol enables receivers to do source authentication on broadcast messages [13]. The use of symmetric algorithms for authentication fails if the secret key is compromised. Asymmetric cryptographic protocols provide secure authentication but are computationally extensive and have high overhead. The TESLA protocol achieves asymmetric performance while employing purely symmetric cryptographic functions by using delayed key exposure [13]. In the TESLA protocol, the sender attaches to each message a message authentication code (MAC) created with a secret key only known to the sender. The receiver buffers the message since it is unable to authenticate it. The sender at a later time reveals the secret key used to create the MAC, so that the receiver can authenticate the message.

The TESLA protocol requires that the nodes in the network be loosely synchronized. That is, a receiver is only required to know the upper bound of the synchronization error between itself and the sender. The protocol uses delayed key (produced from a one-way chain) disclosure to achieve asymmetry by which the receivers are able to verify authentication information but not able to reproduce the authentication information. The TESLA protocol uses four stages: sender setup, receiver bootstrap, sender transmission of authenticated messages, and receiver authentication of broadcast messages. The sender, during the setup, determines the length of the one-way chain, which determines the number of time slots that the chain could be used for. The sender constructs the one-way chain using a one-way function recursively. The sender divides time into intervals of equal duration, each of which is assigned a single value from the one-way chain as shown in Figure 2. The sender also determines the time at which the one-way chain values will be disclosed, which is in the order of few time intervals. The receivers need to be loosely synchronized to the sender and know the schedule for disclosing keys to be able to authenticate messages. During the receiver bootstrap, the receiver establishes the key disclosure schedule by receiving the start time, interval duration, length of one-way chain, key disclosure delay and a key commitment to the chain from the sender. Broadcasting authenticated messages involves appending a MAC corresponding to the time interval on the messages sent during the interval. The key used for creating the MAC remains secret for the entire disclosure delay. The sender reveals the secret key after the disclosure delay elapses and the receivers then authenticate the messages.

As shown in Figure 2, all messages sent during the a given interval are signed using the associated one-way chain value (unknown to the receivers). The one-way chain is used in reverse which means that any value associated with a time interval can be used to derive values for previous intervals but not values used in subsequent time intervals. The ability to retrieve previous keys is a good feature for lossy channels through which encryption keys are sent. The sender creates a MAC using the contents of the message and the one-way chain value associated with that time interval. The MAC is attached to the message along with the most recent one-way chain value that can be disclosed. The format of a message P_j sent in the ith interval is $P_j = \{M_j || MAC(K_i', M_j) || K_{i-d}\}$ where M_j is the message, $MAC(\bullet)$ is the message authentication code and K_{i-d} is the secret used to encrypt messages in the $i - d$ interval.

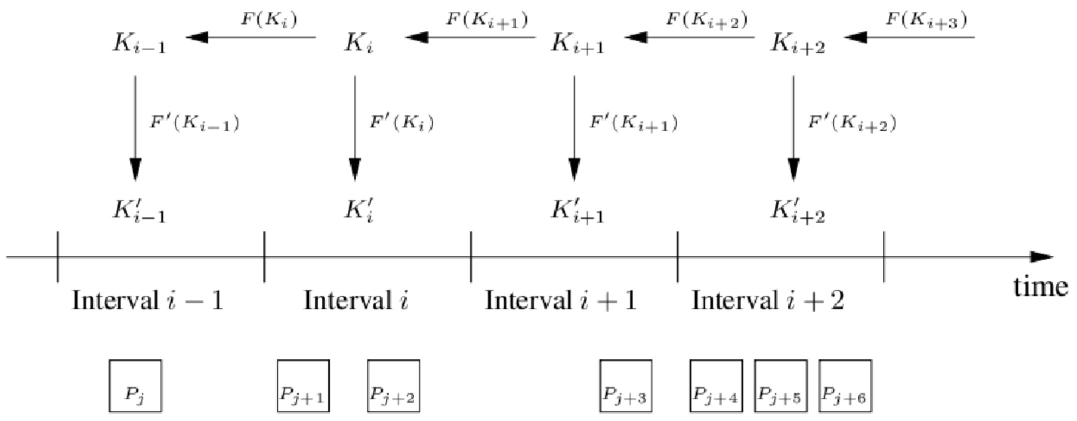


Figure 2: The TESLA protocol Dynamics[13]

The receiver knows the schedule of disclosing the keys and based on synchronized clocks (albeit only loosely) can determine if the key used on the received message is unknown to it. By extrapolating the expected interval that the sender is in, the receiver can determine that the sender could not have reached the time interval for disclosing the key. If the MAC key is still unknown, the receiver buffers the message otherwise it discards it. The receiver then checks if the disclosed key is valid by using self-authentication and previously disclosed keys. The receiver also checks if the MACs of all packets received during the time interval associated with the key are valid. Only messages with valid MACs are accepted, otherwise they are discarded. Any receiver with loose time synchronization and an authentic key chain commitment can authenticate messages but not forge a message with a valid MAC.

The TESLA protocol offers a good solution to the authentication problem over the RDS network. Key distribution is addressed by the use of dynamic keys in the protocol. There are uncertainties about bootstrapping the receivers, which, if done in-band, could possibly allow an attacker to bootstrap all the receivers to an invalid sender. The initial bootstrapping could be done at the time of deployment by a technician installing the PCT (using an out-of-band channel). Attack opportunities arise when all the values of the one-way chain are exhausted and a new chain is created which requires the receivers to be bootstrapped again to distribute chain commitment (and possibly key exposure delay if is different for the new chain). If a PCT is powered off (e.g. battery runs out) or resets, it needs to be bootstrapped again when powered up. An ideal way of bootstrapping the PCTs should be automated to avoid customer involvement. An adversary could learn how bootstrapping is done by eavesdropping which would allow him/her to launch attacks on the PCTs. If the attacker can successfully bootstrap receivers to a bogus key chain, the PCTs will not respond to the legitimate messages. The values of the one-way chain are verified by using previous values because of the recursive way the chain is generated. When a new chain is used, there is need to communicate the key commitment value securely to ensure that the receivers have the correct value. A partial solution to the bootstrapping problem is to send the commitment value in the last time interval(s) of the old chain. The last time intervals of an old chain could be set aside to communicate the bootstrapping information for the next chain to be used. This would solve the problem of key exhaustion, but not of a rebooted (reset) device.

The TESLA protocol also has a possibility of a denial of service attack since the messages are buffered until the key is disclosed. If the disclosure delay is long, an adversary could inject bogus messages into the network and fill up the buffers at the receivers while they are waiting for the key to be disclosed in order to authenticate messages received in a given interval. Such an attack would be to replay a message(s) that is not yet authenticated at the receivers, which would be buffered and cause exhaustion of resources. Until the key for a given interval is exposed, the receivers will buffer all the messages received in a time interval. An attacker could possibly exploit this weakness and cause denial of service.

Appending the MAC and the secret key to the message increases the size of the transmitted message. The RDS networks offers limited bandwidth and initial studies on the RDS suggest that larger messages have a lower chance of reception. The overall size increase of the TESLA protocol should be studied more before deployment in the RDS network. The security concerns of TESLA stated above also need to be studied further and be addressed before implementing TESLA for the PCT system.

5 Cryptographical Security Measures

The following authentication schemes have been identified as good candidates from the literature. The methods described in the previous section offer good authentication solutions but have some drawbacks. The DNP3 solution presented employs a challenge/response method of authentication which cannot be achieved over the RDS network. The RF fingerprints require additional hardware/DSP for fingerprinting capabilities which could increase the cost of receivers. The TESLA protocol suffers from a denial of attack as mentioned previously.

5.1 BiBa Signature Protocol

The BiBa protocol as described in [14], is a general solution that can be applied to sign broadcast data based on one-way functions without trapdoors. The BiBa signature scheme is efficient, robust to packet loss and scales well to a large number of receivers. However, the public keys used in the BiBa protocol are large and the time to generate the signatures is long. For the purpose of the PCT system, the signature generation overhead can be tolerated. We assume that the sender is equipped with powerful computing resources to handle the signature generation overhead. The small signature sizes make the BiBa protocol a good candidate for the PCT system which is to be deployed over a bandwidth constrained network. Moreover, small signature verification overhead allows the end devices (PCT's) to be simple and cheap. The impact of the BiBa broadcast protocol on the RDS network needs to be studied further before deployment.

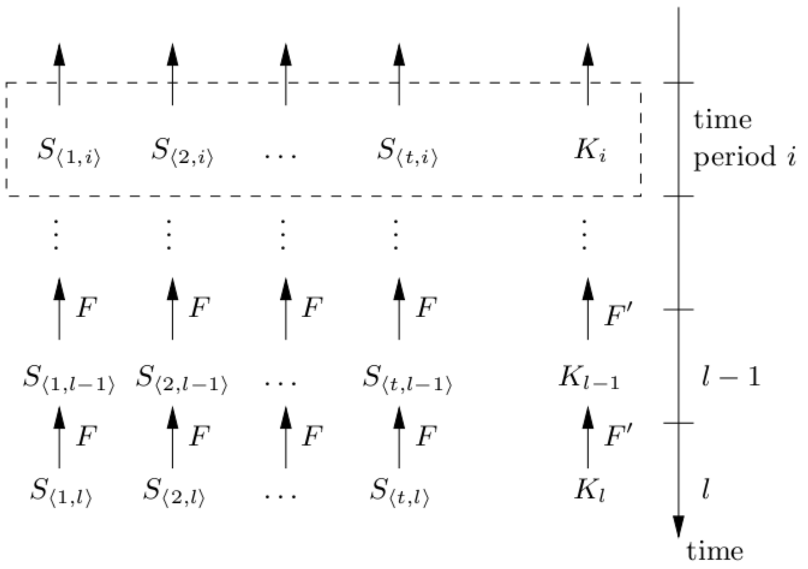


Figure 3: The BiBa Broadcast Protocol dynamics[14]

Figure 3 shows the dynamics of the BiBa broadcast protocol. The sender divides time into periods of equal duration. The sender then creates t chains of *SElf Authenticating vaLues* (SEALs), $S_{<1,i>}, ...S_{<t,i>}$, and a Salt chain, K_i, associated with time interval i. The SEAL and Salt chains are of length l, hence they last l time intervals. The Salt key is used by the sender to create the SEALs and is required for authentication of SEALs at the receiver. The SEALs are generated recursively by applying a pseudo-random function F as follows: $S_{<i,j>} = F_{S_{<i,j+1>}}(K_{j+1})$; for $(1 \leq i \leq t)$ and $(1 \leq j \leq l)$. The use of the Salt key forces an attacker to obtain the pre-image of the Salt chain as a pre-requisite to finding the pre-images of the SEAL chains. Therefore an attacker cannot precompute the SEALs for subsequent time periods without knowledge of the Salt key [14].

At the beginning of each active interval, the sender broadcasts the value of the active Salt (K_i) to the receivers. The dotted box in Figure 3 shows an active time interval with the associated Salt key and SEALs. To sign a message m during the active interval i, the sender creates a hash of the message $h = H(m|c)$, which is used to seed a hash function $G_h()$ used to produce a signature; where c is a counter that is incremented when a signature could not be obtained. The sender uses the hash function $G_h()$ on the t SEALs and observes any k-way collisions from distinct SEALs. That is, $S_{<1,i>} \neq S_{<2,i>} \neq .. \neq S_{<k,i>}$ such that $G_h(S_{<1,i>}) = G_h(S_{<2,i>}).. = G_h(S_{<k,i>})$. The k SEALs that result in a collision form the signature and are then sent together with the message as $(< S_1, ..., S_k > ||m)$. The receiver then authenticates the message if $G_h(S_1) = .. = G_h(S_k)$ and $S_1 \neq .. \neq (S_k)$. During signature generation, it is possible that $G_h()$ applied on all t SEALs fails to produce at least k collisions, in which case a signature cannot be formed. The counter c serves to get a different hash value h in the event that $G_h()$ fails to produce at least k collisions from all t SEALs. The receiver is assumed to know the value k, the hash function H and hash function family G.

The security of the BiBa protocol relies on the fact that a potential attacker knows fewer SEALs than the sender with which to forge a signature. Therefore the sender only reveals the SEALs that are used in creating a signature. The receiver is able to verify that an adversary has a smaller number of SEALs with which to forge a false signature by relying on time synchronization. The BiBa protocol requires loose synchronization between the sender and receiver. When a receiver receives a signed message, it verifies that the sender has not yet revealed r SEALs based on synchronization. If the sender and receivers have a maximum synchronization error of δ, the sender can only send at most $\lfloor r/k \rfloor$ messages within δ time without compromising the security [14]; where r is the maximum number of active SEALs an attacker is allowed to know and k is the number of SEALs revealed in one message. [14] presents a study on how the BiBa protocol can be used in an application and how to determine the BiBa protocol parameters. A receiver is bootstrapped to the sender by revealing all the SEALs and Salt key from one active interval so that subsequent SEALs can be verified. During receiver bootstrapping, the receiver receives the initial values of the SEAL and Salt chains. The receiver then commits to the chains, which allows verification of subsequent SEAL and Salt values. The bootstrap information, which consists of the initial values of the SEAL and Salt salt chains (i.e. the commitment keys of the SEAL chains and the Salt chain), is referred to as the public key in this document. There are extensions that allow efficient bootstrapping of receivers in [14] by periodically sending the SEALs of a time period. The receivers then use the information to verify subsequent SEALs that are used to sign messages.

5.1.1 Authenticating PCT Messages using BiBa

The authentication of messages in the PCT system using the BiBa protocol involves a tiered solution. A long-term BiBa instance is used to send short-term BiBa instance commitment keys which serve as public keys. The long-term BiBa instance is conceptually designed to last the entire lifetime of the PCT system. Multiple levels of BiBa instances can be used as necessary to prolong the lifetime of the long-term BiBa instance. This report uses only two levels to demonstrate the concept and evaluate the performance. Extensions to multiple levels can be done easily following the definition presented here. The long-term BiBa instance is made up of long SEAL chains with large SEAL sizes, hence it is more secure and has a large public key (commitment key). The long-term BiBa instance is used infrequently to bootstrap the receivers to new short-term BiBa instances. The short-term SEAL chains are used to authenticate the application messages using BiBa signatures.

Figure 4 shows the dynamics of our authentication construct using the BiBa one-time signature and broadcast protocol. Initially the sender creates a long-term BiBa instance by following the construct described above. The long-term BiBa commitment keys which serve as a public key are then communicated to the receiver(s). The receiver(s) saves the commitment keys to authenticate subsequent messages signed by the long-term BiBa instance. The long-term BiBa instance should be bootstrapped offline or at the time of installation in the case of the PCT system. To allow recovery in the event that the receiver is rebooted, the commitment chain is stored in non-volatile memory. A receiver that is shut down for long periods can synchronize to the short-term BiBa instances by receiving the periodic short-term BiBa instance commitment chains signed by the long-term BiBa instance. The only requirement is that such a device retains the initial commitment key of the long-term BiBa instance.

An example illustrating how the protocol works is presented below:

- The receiver commits to the long-term BiBa instance, shown by label A in Figure 4. In the PCT system this could be done offline at the time of installation. A technician or home owner keys in the commitment key of the long-term BiBa instance, which is then saved into non-volatile memory on the device. With the long-term BiBa instance commitment keys, the device can authenticate short-term BiBa instance commitment keys signed using the long-term BiBa instance.

- The home device receives the periodic short-term BiBa instance commitment information signed using the long-term BiBa instance, shown by label B in Figure 4. The receiver can authenticate the commitment keys of a short-term BiBa instance as described above. When the authentication is successful, the receiver commits to the short-term BiBa instance. The short-term BiBa instance is used to authenticate application messages.

- The home device receives application messages signed using the short-term BiBa instance (shown by label C in Figure 4). The application messages are authenticated as described in the definition of the BiBa protocol.

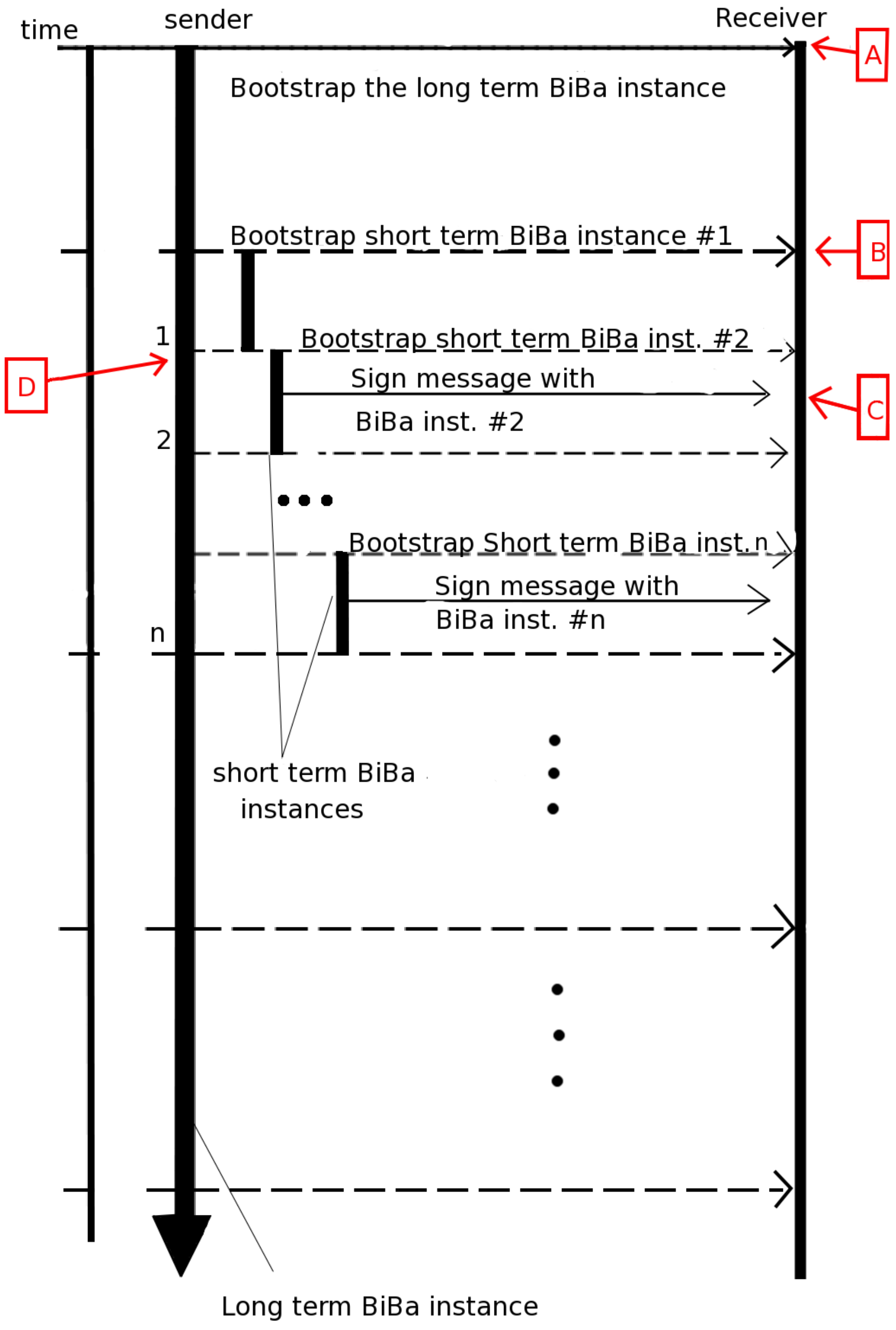

time
sender
Receiver
A
Bootstrap the long term BiBa instance
Bootstrap short term BiBa instance #1
B
1
D
Bootstrap short term BiBa inst. #2
Sign message with
BiBa inst. #2
C
2
Bootstrap Short term BiBa inst. n
Sign message with
BiBa inst. #n
n
short term BiBa
instances
Long term BiBa instance

Figure 4: Using the BiBa signature to sign messages

- The short-term BiBa instance expires after l time intervals elapses. Then the long-term BiBa instance creates a new short-term BiBa instance and sends the commitment key to the receivers (illustrated by label D in Figure 4).

5.1.2 Protocol Messages

Figure 5 shows the structure of the messages sent by the BiBa protocol. A description of the structure of the protocol messages sent to facilitate authentication using BiBa instances is given below. Reference is made to Figure 5 to describe the different fields of the messages.

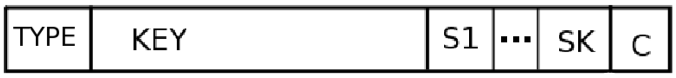


Figure 5: Structure of BiBa messages

TYPE: Describes the type of data that is carried in the message.
0 : Application messages are carried in the KEY field
1 : Short-term BiBa instance commitment key is carried in the KEY field
2 : A Salt key is carried in the KEY field
3 : Long-term BiBa commitment key. This option is not used if the long-term BiBa instance commitment is done off-line

KEY: The Data that is being sent in the message which is signed. Depending on the value of the TYPE field it can either be a message sent by the application or Salt key to be signed by the short-term BiBa instance, or short-term BiBa commitment key.

S1...SK: Part of the signature formed by the k SEALs that resulted in a collision. The size depends on the value of k.

C: The counter that is incremented when a signature is not obtained, which is part of the signature

Figure 6 shows the actions applied to application messages as they traverse through the different layers. The reverse operation is performed at the receiver. The application generates messages as described in the Title-24 specification. The messages are then delivered to the System operator who encrypts them for authentication purposes and sends them over the RDS network. The messages are sent over the RDS network as type-11A groups. Each RDS group can only carry 4 bytes of data, so the message is fragmented into multiple RDS groups and sent over the network. The receiver reconstructs the messages from the multiple RDS groups and sends it up the protocol stack to the security layer. The security layer then authenticates the messages and present them to the application layer if the authentication is successful.

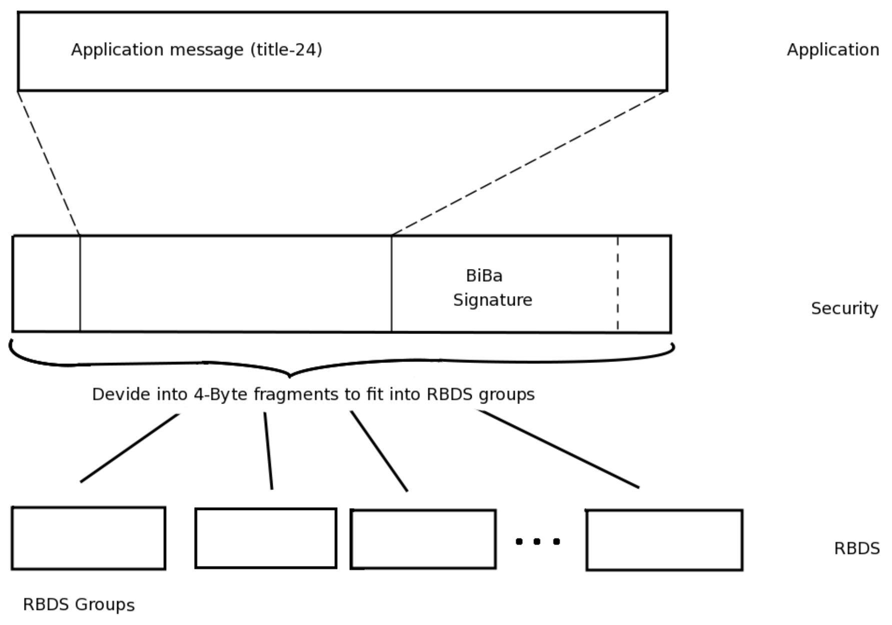

Application message (title-24)
Application
BiBa
Signature
Security
Devide into 4-Byte fragments to fit into RBDS groups
RBDS
RBDS Groups

Figure 6: Operations on the application messages

5.1.3 Setting the BiBa Parameters

The parameters of the security layer are based on approximated application data rate. A BiBa instance with 1024 SEAL chains ($t = 1024$), using 4-way collisions ($k = 4$) can be used to sign 25 messages ($\nu = \lfloor \frac{t\gamma}{k} \rfloor$), with $\gamma = 0.10$; where γ is the fraction of SEALs that can be revealed to an adversary without compromising the security of the protocol (typically $\gamma = 10\%$ [14]). An adversary is only allowed to learn r SEALs from one active period; where $r = t\gamma$. Each signature reveals k SEALs to the adversary, hence only ($\nu = \lfloor \frac{t\gamma}{k} \rfloor$) messages can be signed within a single time interval. An adversary who knows r SEALs needs to make 2^{35} computations to forge a valid signature of a BiBa instance with the above parameters according to [14]. If we assume that the application sends an average of 20 event messages every day, a single time interval for the short-term BiBa instance is sufficient to authenticate an entire day's messages. Consequently, a short-term BiBa instance with SEAL chain lengths of 50 ($l = 50$), can be used for 50 days before it expires. If the long-term BiBa instance is designed with the same parameters as the short-term instances (i.e. $t = 1024$, $k = 4$, $\gamma = 10\%$), then it can be used to commit 25 short-term BiBa instances in a single time interval. A single time interval for the long-term BiBa instance can then be made to last up to 1250 days (3.4 years). The entire long-term BiBa instance will then last 171 years.

5.2 HORSE Authentication Protocol

The HORSE authentication protocol extends the HORS (Hash to Obtain Random Subsets) protocol which is an extension of the BiBa protocol to provide broadcast authentication. HORSE and BiBa are r-time signature schemes that provide unforgeable signatures which can be verified by using publicly available information. R-time signature schemes achieve faster signature generation at the expense of larger key sizes. Generally, the generation of such signatures is faster than public-key signatures but they can only be used to sign r messages [15].

The HORS protocol works by mapping a message m to a k-element subset of t-element set T. The mapping of a message m is achieved by a collision-resistant hash function H (eg. MD5 or SHA-1). Then, for messages m_1 and m_2; $m_1 \neq m_2$, it should be impossible to get $H(m_1) \subseteq H(m_2)$. In a general case for r messages $m_1, m_2, ..., m_r$, it must be infeasible to obtain $H(m_r) \subseteq \bigcup_{i=1}^{r-1} H(m_i)$. To obtain the k-element subset, the output of the hash function $H(m)$ is split into k substrings each $\log_2(t)$ bits. The substrings are then interpreted as integers j_i, $(1 \leq i \leq k)$, which selects k values in set T. The k values selected from T form the signature $(s_{j_1}, s_{j_2}, ..., s_{j_k})$.

To sign a message m in HORS, the sender initially selects values t and k such that $k \log_2 t \leq |H(\cdot)|^2$. The function H as described above is a collision resistant hash function that maps a message m to k-element subsets of T. The sender then generates the secret key, SK $= (s_1, s_2, ..., s_t)$ by randomly generating t l-bit values. The public key is then, PK $= (v_1, v_2, ..., v_t)$ with $v_i = f(s_i)$,$1 \leq i \leq t$,where f is a one way function. The sender computes $h = H(m)$, and splits h into k sub-strings each of length $log_2 t$ bits. Each substring is interpreted as an integer j_i for $(1 \leq i \leq k)$. The signature is then made of the subset of SK, $(s_{j_1}, s_{j_2}, ..., s_{j_k})$ and is sent along with the message m. To verify a signature $(s'_1, s'_2, ..., s'_k)$ at the receiver, the receiver computes $h = H(m)$. The receiver then splits h into k substrings and interprets the substrings as integers of $log_2 t$ bits. Then it verifies that $v_i = f(s'_i)$, for $1 \leq i \leq k$ otherwise the signature is rejected.

HORSE extends HORS by using one way chains to generate and update the secret key and public key pair. In the HORS protocol one can only sign r messages without losing security. HORSE uses a one way hash function $H()$ to generate chains of values each d values long. To initialize, the sender generates t random values $(s_{<0,1>}, s_{<0,2>}, ..., s_{<0,t>})$ and uses them to construct t chains of length d. The hash function is used recursively d times on each of the t initial values to get a chain as shown in Figure 7. The keys are then used in reverse order of generation. That is, the initial secret key is given by SK_0 $= (s_1, s_2, ..., s_t) = (s_{<d-1,1>}, s_{<d-1,2>}, ..., s_{<d-1,t>})$, where $s_{<i,j>} = H^i(s_{<0,j>})$. The initial public key is given by $PK_0 = (v_1, v_2, ..., v_t)$; $v_i = f(s_i)$, $\forall s_i \in SK_0$

The signature generation and verification is computed as described above for HORS. The secret-key gets updated after each signature is generated. The values used to generate the signature gets replaced by the values preceding them in the respective chains as shown by Figure 8. The figure depicts a scenario where the secret key SK_i gets updated after using the values $s_{<\mu,\alpha>}$, $s_{<\nu,\beta>}$, and $s_{<\xi,\kappa>}$ in a signature. The secret key is then updated to SK_{i-1} with $s_{<\mu,\alpha>}$, $s_{<\nu,\beta>}$, and $s_{<\xi,\kappa>}$ replaced by $s_{<\mu-1,\alpha>}$, $s_{<\nu-1,\beta>}$, and $s_{<\xi-1,\kappa>}$ respectively, while the other values remain unchanged.

$$
\begin{array}{|ccccc|c|}
\hline
s_{\langle 0,1\rangle} & s_{\langle 0,2\rangle} & \cdots & s_{\langle 0,t-1\rangle} & s_{\langle 0,t\rangle} & \\
\downarrow \mathcal{H}(\cdot) & \downarrow \mathcal{H}(\cdot) & \cdots & \downarrow \mathcal{H}(\cdot) & \downarrow \mathcal{H}(\cdot) & \\
s_{\langle 1,1\rangle} & s_{\langle 1,2\rangle} & \cdots & s_{\langle 1,t-1\rangle} & s_{\langle 1,t\rangle} & \\
\downarrow \mathcal{H}(\cdot) & \downarrow \mathcal{H}(\cdot) & \cdots & \downarrow \mathcal{H}(\cdot) & \downarrow \mathcal{H}(\cdot) & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \\
\hline
s_{\langle d-1,1\rangle} & s_{\langle d-1,2\rangle} & \cdots & s_{\langle d-1,t-1\rangle} & s_{\langle d-1,t\rangle} & SK_0 \\
\hline
\downarrow \mathcal{H}(\cdot) & \downarrow \mathcal{H}(\cdot) & \cdots & \downarrow \mathcal{H}(\cdot) & \downarrow \mathcal{H}(\cdot) & \\
\hline
s_{\langle d,1\rangle} & s_{\langle d,2\rangle} & \cdots & s_{\langle d,t-1\rangle} & s_{\langle d,t\rangle} & PK_0 \\
\hline
\end{array}
$$

Figure 7: HORSE protocol

$$
\begin{array}{rl}
SK_i = & \left(s_{\langle\zeta,1\rangle}, s_{\langle\eta,2\rangle}, \ldots, \; s_{\langle\mu,\alpha\rangle}, \; \ldots, \; s_{\langle\nu,\beta\rangle}, \; \ldots, \; s_{\langle\xi,\kappa\rangle}, \; \ldots, s_{\langle\rho,t\rangle}\right) \\
& \qquad\qquad\quad \downarrow \mathcal{H}^{-1}(\cdot) \quad \downarrow \mathcal{H}^{-1}(\cdot) \quad \downarrow \mathcal{H}^{-1}(\cdot) \\
SK_{i+1} = & \left(s_{\langle\zeta,1\rangle}, s_{\langle\eta,2\rangle}, \ldots, s_{\langle\mu-1,\alpha\rangle}, \ldots, s_{\langle\nu-1,\beta\rangle}, \ldots, s_{\langle\xi-1,\kappa\rangle}, \ldots, s_{\langle\rho,t\rangle}\right)
\end{array}
$$

Figure 8: Updating the public and secret keys in HORSE

The receiver updates the public key each time it receives a signed message. The receiver verifies the signature by performing a hash operation on the values that make up the signature and compares them to the public key as described earlier. After successful verification, the receiver updates the public key by replacing the values in the public key that are preceded by the values that make up the received signature. In the example shown in Figure 8, the values that make up the signature ($s_{<\mu,\alpha>}$, $s_{<\nu,\beta>}$,$s_{<\xi,\kappa>}$) replace the corresponding values in the public key (v_α, v_β,v_κ) in the public key. In a lossy environment, the receiver may not successfully receive the signed message and lose synchronization, which would lead to unsuccessful signature verification. To avoid the loss of synchronization, the index corresponding to the position of the values that make up the signature in their corresponding chains is sent as part of the signature. This lets the receiver know how many hash operations it needs to perform to verify each value in the signature. The signature is then formed by ($< \alpha_1, s_1 >$, $< \alpha_2, s_k >$,$< \alpha_k, s_k >$) where $\alpha_i \in [0, d-1]$ gives the position of s_i in the chain. The signature is then verified if $H^{d-\alpha_i}(s_i) = s_{<d,i>}$ for $1 \leq i \leq k$.

In the worst case, the maximum number of messages that can be signed is d. That would happen if at least one chain gets used to create a signature every time a message is sent. Based on probability, the expected number of messages that can be signed is $d/(1-e^{-k/t})$ [15]. As an example, in [15] HORSE is expected to sign up to $65 \cdot d$ messages compared to 4 messages for HORS, for the same parameters $t = 1024$ and $k = 16$. The tradeoff is that the memory required to store the chain values in HORSE is d times that of HORS. Alternatively, if memory is not enough, HORSE could require up to $k \cdot d$ hash evaluations to generate each signature. However, [15] mentions a technique that allows efficient storage of chain values that requires only storing $\log_2 d$ hash values and performs at most $\log_2 d$ hash evaluations per step.

5.2.1 Authenticating PCT Messages using HORSE

Each HORSE instance is expected to sign $n = d/(1 - e^{-k/t})$ messages on average as explained above. To sign messages exceeding $d/(1 - e^{-k/t})$, there is a need to use a new HORSE instance after the current one expires. To address the issue of signing messages exceeding $d/(1-e^{-k/t})$ messages, we propose a tired solution similar to the one employed for using BiBa to sign PCT messages. A long-term HORSE instance is used to send the initial public key of a new short-term HORSE instance when the current one expires. The initial public keys of the short-term HORSE instance are signed by the long-term HORSE instance and sent to the receivers to allow the receivers to verify subsequent messages. Figure 9 shows the structure of the our construct to provide authentication for the PCT system.

An example to illustrate how the HORSE construct works is presented below:

- The initial public key of the long-term HORSE instance is sent to the receivers at the time of installation. This is done by a technician installing the PCTs, shown in Figure 9 by label A.
- Short-term HORSE initial public keys are then sent to the receivers signed using the long-term HORSE private key as described above. Label B in Figure 9 shows a short-term initial public key being sent to the receivers.
- Application messages are signed with the short-term HORSE instance as described previously, shown by label C in Figure 9.
- Each short-term instance HORSE on average will send $d/(1-e^{-k/t})$ messages after which a new short-term HORSE instance needs to be created. When one of the chains in the short-term HORSE instance is about to be exhausted (left with say 3 values), a new short-term HORSE instance is created and the public key is sent to the receivers. When the first chain is exhausted (left with 1 value), a message is sent to the receivers to instruct them to use the last public key they received. The message sent to the receiver to switch to the new public key is encrypted with the expiring short-term HORSE instance, such that the chain that had 1 value left have all its values used.

A short-term HORSE with $t = 1024$, $k = 4$, and $d = 50$ on average will sign 12825 messages before a new short-term instance is required. If the long-term HORSE instance has the same parameters, then 2565 short-term HORSE instances can be signed. Therefore on average $12825^2 = 164480625$ application messages can be signed. Keeping the previous application data rate of 20 messages per day, the construct can last 8224031 days (22531 years).

5.2.2 HORSE Protocol Messages

The protocol messages that are communicated to facilitate the use of HORSE for authentication in our solution follow the format used for BiBa (See Section 5.1.2). The structure of the messages is shown in Figure 10. The message fields are defined as follows:

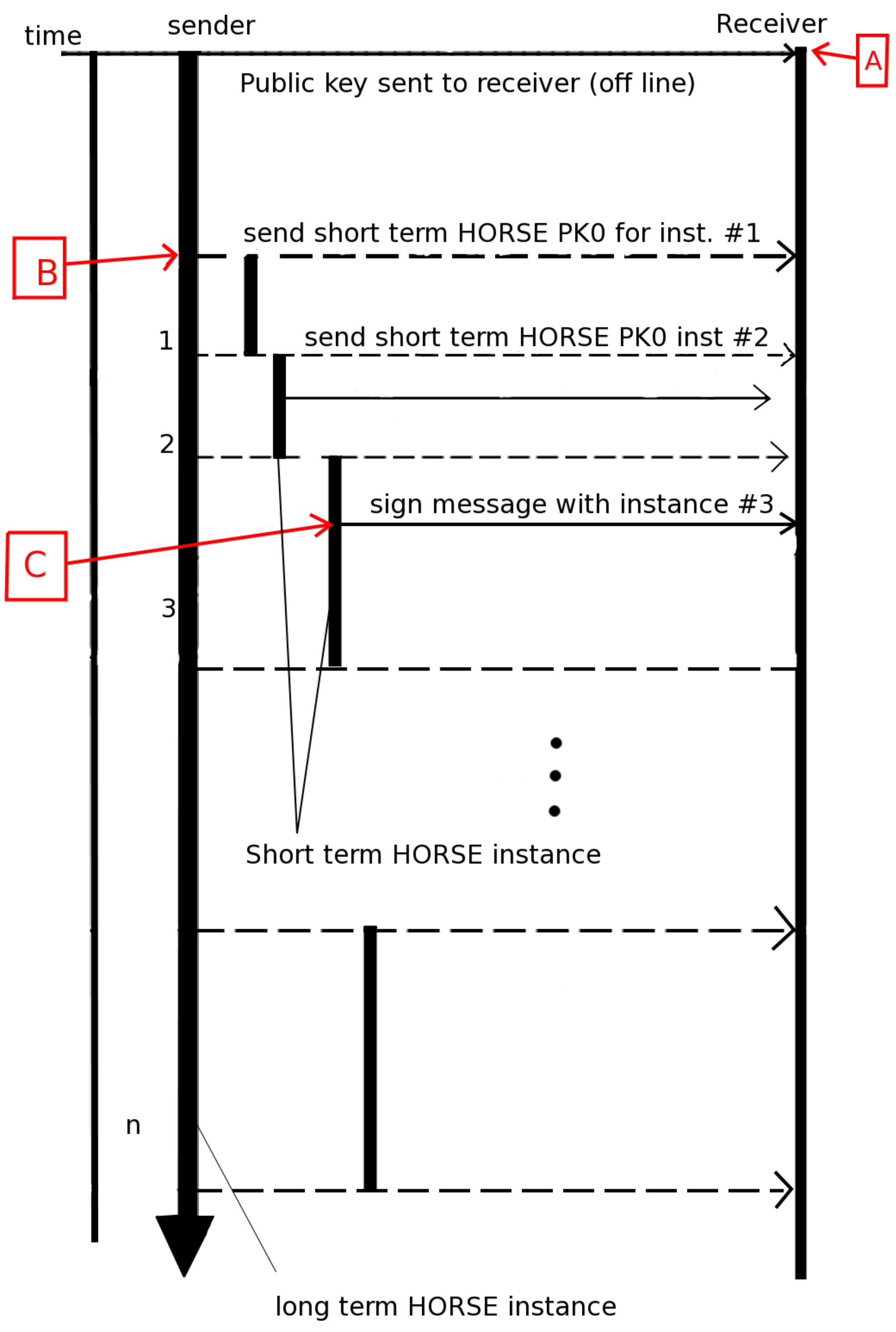

time
sender
Receiver
A
Public key sent to receiver (off line)
send short term HORSE PK0 for inst. #1
B
1
send short term HORSE PK0 inst #2
2
sign message with instance #3
C
3
Short term HORSE instance
n
long term HORSE instance

Figure 9: Employing HORSE to authenticate messages

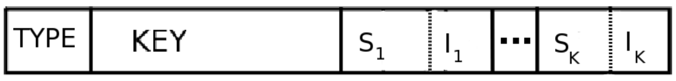


Figure 10: Structure of HORSE protocol messages

TYPE: Describes the type of data that is carried in the message.
0 : Application messages are carried in the KEY field
1 : Short-term HORSE instance public key is carried in the KEY field
2 : Command to switch to newly received public key
3 : Long-term HORSE commitment key. This option is not used if the long-term HORSE instance commitment is done off-line

KEY: The Data that is being sent in the message which is signed. Depending on the value of the TYPE field it can either be a message sent by the application or a command to switch to the next HORSE instance, or short-term HORSE public key.

$< S_1, I_1 >$...$< S_K, I_k >$: The signature formed by the k values that forms the subset (S_i), and the index of the values in the chains(I_i); with $1 \leq i \leq k$. The size depends on the value of k.

The flow of messages will follow the same diagram as depicted by Figure 6

5.3 The Elliptic Curve Digital Signature Algorithm

Elliptic Curve Digital Signature Algorithm (ECDSA) is similar to the Digital Signature Algorithm (DSA), but employs elliptic curves over a finite field[16]. Elliptic curve cryptography offers faster verification and smaller keys for equivalent security with other public key systems [17]. Based on the complexity of the Elliptic Curve Discrete Logarithmic Problem, it is computationally infeasible to forge a signature if appropriate parameters are employed.

A finite field F is made up of a finite number of elements together with two binary operations on F. The binary operations, addition and multiplication have special arithmetic properties as defined in [16]. The order of a finite field is the number of elements in the field. If p is a prime number, then the field F_p is called a prime field and is made up of integers $\{0,1,2,...,p-1\}$. Addition and multiplication of elements of F_p are done modulo p. That is $a+b=r$; where $r=(a+b)\mathbf{mod}\ p$, and $a \cdot b = s$;where $s = a \cdot b\mathbf{mod}\ p$. An elliptic curve E on a finite field F_p; where $p > 3$ is an odd prime, is given by the equation:

$$y^2 = x^3 + ax^2 + b \tag{1}$$

where p is a prime number, $a, b \in F_p$, and $4a^3 + 27b^2 \neq 0(\mathbf{mod}\ p)$. The set $E(F_p)$ consists of all points (x, y) $(x, y \in F_p)$ that satisfy equation 1 and a point ϑ located at infinity. The point ϑ is the identity element of the group.

All the elements of the group have the properties:

$$P + (-P) = (-P) + P = \vartheta$$

and

$$P + \vartheta = \vartheta + P = P$$

The security of elliptic curve cryptography comes from the Elliptic Curve Discrete Logarithmic Problem (ECDLP). The ECDLP consists of finding a value k such that $P = kQ$ given P and Q (with $P, Q \in F_p$). There is no efficient known algorithm that can compute the value of k [17]. The parameter requirements to achieve resilience to known attacks are outlined in [16]. [16] also gives ways of generating cryptographically secure parameters for elliptic curves using several methods.

To sign a message, initially the sender and receiver agree on an elliptic curve with a base point P over the field F_p. The sender has a private key x and a public key $Q = xP$. The parameters of the curve a, b, P, q, F_p as well as the public key Q are assumed known to the receiver. To sign a message m:

1. The sender generates a random number k; $k \in [1, n-1]$ and then computes $kP = (x_1, y_1)$. The value x_1 is then converted to an integer $\bar{x_1}$
2. Compute $r = \bar{x_1}\mathbf{mod}\ n$. If $r = 0$ then step 1 is repeated until $r \neq 0$.
3. Compute $k^{-1}\mathbf{mod}\ n$.
4. Compute SHA-1(m) and convert the output string into an integer e.
5. Compute $s = k^{-1}(e + dr)\mathbf{mod}\ n$; with $s \neq 0$. If $s = 0$ then go back to step 1.

6. The pair r,s form the signature and sent to the receiver.

When the receiver receives the signed message it performs the following steps to verify the signature.

1. Confirm that $r, s \in [1, n-1]$.
2. Compute SHA-1(m) and convert the output string into an integer e.
3. Compute $w = s^{-1}\mathbf{mod}\ n$.
4. Compute $u_1 = ew\mathbf{mod}\ n$ and $u_2 = rw\mathbf{mod}\ n$.
5. Compute $X = u_1P + u_2Q$. If $X = \vartheta$ reject the signature.
6. Covert the x coordinate of X to an integer $\bar{x_1'}$, and compute $v = \bar{x_1'}\mathbf{mod}\ n$
7. Accept the signature if $v = r$

5.3.1 Authenticating PCT Messages using ECDSA

Authenticating PCT messages does not require a lot of changes to ECDSA. The use of the hash function SHA-1 in step 2 of signature generation can be omitted since the PCT messages are small. Consequentially step 2 at the receiver will also be omitted resulting in less computations. The use of hash function is to obtain a fixed length string from variable message lengths . In practice the messages could be a file a few kilo bytes long, hashing it using SHA-1 results in a string 160 bits long.

The value $n > 2^{160}$ is recommended to protect against attacks as outlined by [16]. The sizes of the signature (r,s) is dependent on the value n, $r, s \in [1, n-1]$. Taking a minimalist approach and using $n = 2^{160}$, we have both r and s being 20 bytes long (the signature is 40 bytes long). The above parameters can then be used for the PCT system for authentication purposes. It may be necessary to use a dynamic approach where the elliptic curve parameters used are refreshed periodically. To allow such a construct, the sender uses one long-term secret key to sign the new parameters when sending to them to the receivers.

6 Simulation Results and Analysis

The evaluation of the protocols was done using the Network Simulator tool (NS-2.30). The evaluation consists of building a model of the RDS network. The physical layer propagation model was designed to match closely real conditions by conducting field measurements to calibrate the model. Signal strength readings of FM broadcasts were taken from various places in Ottawa and used to characterize the channel gain. The locations for signal strength measurements were chosen randomly and are shown in Figure 11. Readings were taken from two FM broadcast channels, Hot 89.9 (operation at 89.9 MHz) and Bob FM (operationg at 93.9 MHz). Both stations have their transmitting antennas in Camp Fortune shown by the top left tag on Figure 11. Hot 89.9 transmits at an effective radiated power (ERP) of 27 KW while Bob FM has an ERP of 95 kW. The signal strength measurements are presented in Figure 12 by the crosses (red). All the readings were taken from indoors to reflect operation conditions for PCTs. The variations of the signal strength readings as shown in Figure 12, are due to the different conditions at the measurement sites. Some of the locations were on high rise apartment buildings with potential line of sight conditions while others were in basement apartments. Other sites were in town houses with dry wall and wooden walls while others had concrete walls. All these different conditions have effects on the signal propagation and hence the signal readings were diverse although they were taken from distances relatively similar from the transmission antenna. Weather conditions and other unknown environmental effects also account for the vast differences in the readings. To account for the diverse nature of the data, two approximations were made to reflect the best case and worst cases. Figure 12 also shows the estimated signal strength obtained through simulations. The physical layer signal propagation model used accounts for large scale fading channel by using the Shadowing model provided in NS-2. The small scale channel fading was modeled by using a Ricean model obtained from [18].Two pathloss exponents were used to account for the diverse data as shown in Figure 12. Taking a conservative approach the evaluation was conducted for the worst case scenario hence the lower curve on Figure 12 was used to evaluate performance.

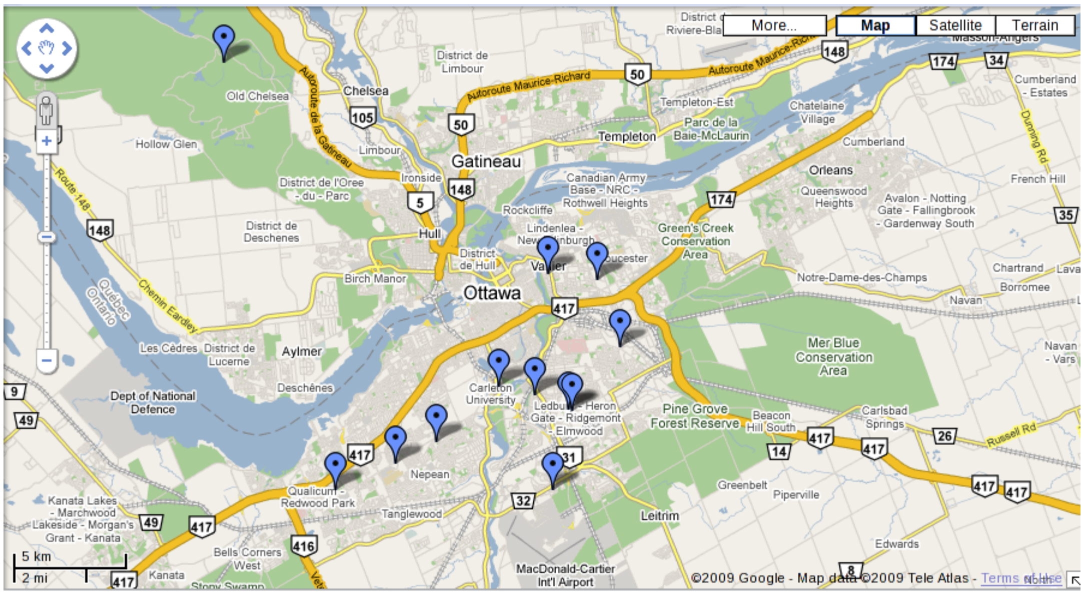


Figure 11: Locations of signal strength measurements

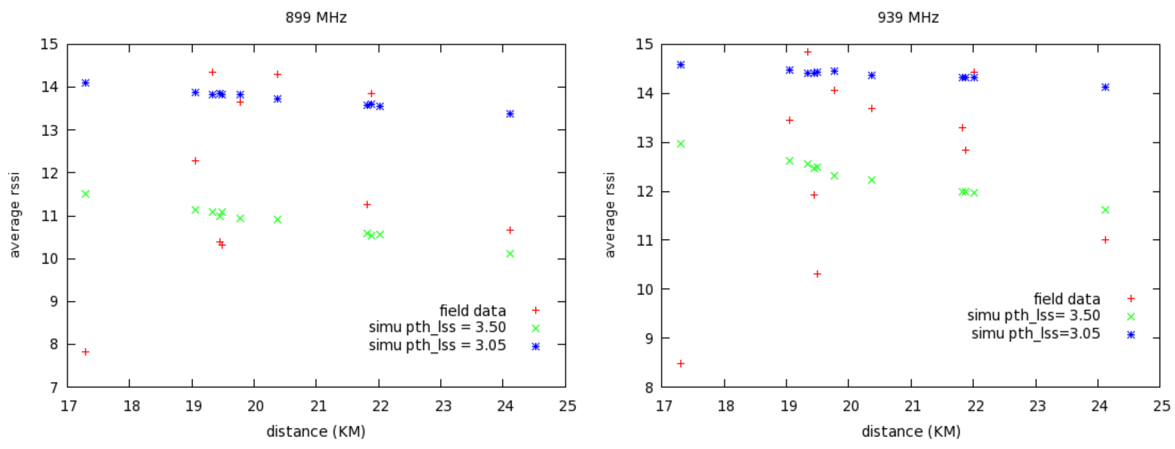


Figure 12: Estimation of the communication channel using field data

The media access of the model was designed to closely resemble the RDS network. That is, the fragmenting of messages into RDS groups was done to model the real network. Security is provided for by implementing the above the three authentication schemes in the link layer. Table 1 shows the parameters of the RDS network used to evaluate the security protocol. All results presented here, unless stated otherwise are based on the parameters in Table 1.

RDS Mode	Real-Time
Number of re-transmissions	5
Raw data bitrate	1187.5 bps
Propagation model pathloss exponent	3.50
Transmission power	27KW
Application Message size	30 Bytes

Table 1: Physical network parameters

6.1 BiBa Performance Results

From the simulations of the security protocol, a BiBa instance of 1024 SEAL chains was found to take 440 seconds (7.33 minutes) to bootstrap, assuming that no other application uses the network. Reducing the number of SEAL chains by half reduces the bootstrapping time by half. Alternatively, SEALs of smaller size could be used to reduce the size of the public key. The time taken to bootstrap a new BiBa instance needs to be short to avoid periods where the receiver is not synchronized with the sender. If such periods are allowed, the receiver will be unable to authenticate the new messages. To avoid such a problem, the last few time intervals (depending on the size of the short-term BiBa commitment key) of the current short-term BiBa instance could be used to bootstrap the next short-term BiBa instance.

The commitment keys used to bootstrap BiBa instances (public keys) are in the order of a few kilo bytes. Successful reception of such messages over the lossy RDS channel becomes a problem. Initial studies of the RDS network show that the probability of receiving messages sent over the network is inversely proportional to the size of the message. The probability of receiving a message made up of m RDS groups and retransmitted n times is given by $P = (1 - P(fail)^n)^m$; where $P(fail)$ is the probability that a single RDS group is not received. The size of the BiBa commitment key using 512 SEAL chains ($t = 512$), with each SEAL 16-bits (2 Bytes) long for the short-term BiBa instance, the key is $m = \frac{512*2}{RDS\_Group\_size} = 256$ groups; with ($RDS\_Group\_size = 4$). Such a large value for m diminishes the probability of reception rapidly.

The trick mentioned earlier of avoiding unsynchronized periods between successive BiBa instances helps to increase the probability of receiving a new public key to an already synchronized node. This is because such a node effectively has 2 chances of receiving a the commitment key for the next short-term BiBa instance. The first chance comes from using the long-term BiBa instance to bootstrap the new short-term BiBa instance. The second chance comes from using the short-term BiBa instance to avoid unsynchronized periods between successive short-term BiBa instances. To increase the chances of bootstrapping short-term BiBa instances the number of times a public key is sent can be increased by sending commitment keys by both the long-term and short-term BiBa instances.

Different values for the number of SEALs in the short-term BiBa instance (t) have an impact on the reception of application messages. Figure 13 shows how the probability of receiving application messages vary with different commitment key sizes. The results presented in Figure 13 are for fixed SEAL sizes (16 bits). A larger value for t results in a larger commitment key which has lower probability of reception. It can be seen that generally a larger value of t results in lower reception probability compared to smaller values of t.

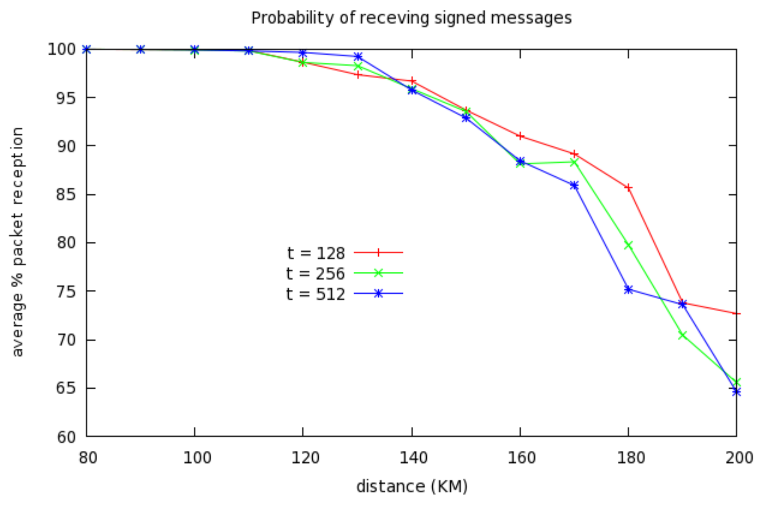


Figure 13: The effects of different of public key sizes on reception of messages

The reception of application messages depends largely on the successful bootstrapping of the BiBa instances at the receiver. When the receiver fails to bootstrap, all the messages received during that period will not be successfully verified. The results depicted in Figure 14 show degraded performance against the case with no security because of the obvious bootstrapping problem caused by large commitment keys. More application messages are rejected by the security layer caused by unsynchronised receivers when bootstrap information is not received. The successful reception of messages varies accordingly with the number of retransmissions of RDS groups. By increasing the number of RDS re-transmissions, the reception of the individual messages and more importantly the commitment keys, will be increased.

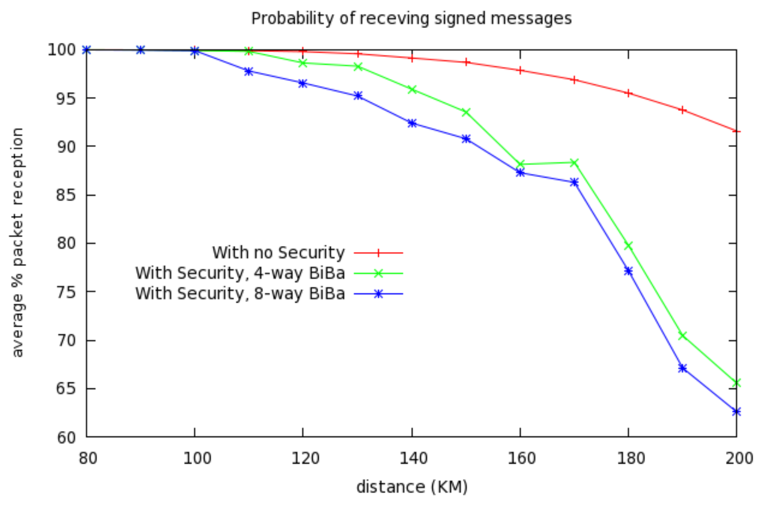


Figure 14: The effects of different signature sizes on reception of messages

The signature sizes are linear in k for a k-way BiBa signature, and public keys are linear in the number of SEALs (t). The signed messages are bigger than unsigned messages because of the signatures. Increasing the number of collisions required for the signature (k) increases the sizes of the transmitted message but also increases the security of the BiBa protocol by lowering the chances of an adversary to successfully forge a signature. It is expected that messages bearing larger signatures incur lower reception at the receiver based on initial studies of the RDS network. Figure 14 shows the probability of receiving messages using different values of k. The graph on top shows the probability of reception of messages with no authentication. As shown by Figure 14, the messages of an 8-way BiBa signature scheme are less likely to be received than those of a 4-way BiBa scheme. The difference between the signed messages and unauthenticated messages is large because of the dependence on successful bootstrapping of receivers for signed messages. The successful bootstrapping of receivers diminishes fast with distance and effectively reduces the coverage area that a Systems Operator can offer high availability of services.

The signatures introduce significant overhead to messages if the messages are small. An 8-way BiBa signature scheme with 16-bit SEALs has a 100% overhead on 16-Byte messages without considering the bootstrap keys. The messages that are transmitted in the PCT system are very small messages, in the order of tens of bytes. Therefore the signature overhead is large and together with the communication of public keys, the BiBa protocol as used in our design is not bandwidth efficient. However, if the SEAL chains are long and each BiBa instance lasts for long periods of time, the consumption of bandwidth for security reasons can be very small. If one commitment key is sent every 24 hours to bootstrap the receivers, then the bandwidth consumption by background traffic is small. For a short-term BiBa instance with 512 SEAL chains, each SEAL 2 Bytes long, the bandwidth consumption will be $\frac{512*2*8}{24*60*60} = 0.0948bps$. With this construct, the availability of the service would be increased by multiple transmissions of the commitment keys for the short-term BiBa instance.

The operation of the RDS network calls for careful design of the interaction of the security layer and other applications running on top of the RDS network. From simulations there are instances when the application messages can pre-empt the transmission of messages when RDS is operating in Real-Time Mode. If such an event occurs while the short-term BiBa commitment keys are being transmitted, the security protocol would perform badly. Therefore, both the application using the security protocol and other applications running on top of RDS need to be designed to avoid pre-empting of the security commitment keys.

6.2 HORSE Performance Results

Simulation conditions for HORSE were kept equivalent for the case employing BiBa as the security protocol. To be specific, the memory required to store the one way chains at the sender was kept constant. Therefore, the number of chains, chain value sizes and chain lengths were kept equal for both cases. Fixing the mentioned parameters results in an equivalent size of the public keys for HORSE and BiBa. To improve the chances of successful bootstrapping to new HORSE instances, the initial public keys could be sent periodically, the same as described for BiBa.

The size of the signature in HORSE is larger than in the case where BiBa is employed. The increase in signature size is a result of including the position of the values that make up the signature in their respective chains. The signature size in bits is $\lceil log_2 d \rceil \cdot k$. Therefore, the signature size increases linearly with the number of values that make up the signature,(k), and logarithmically with the lengths of the chains used to generate the signatures (d). There is a tradeoff between the lengths of the chains d, and the storage requirements and signature size. A large value of d results in a higher number of messages that can be signed by a single HORSE instance. On the other hand, a large d requires more storage and/or computation by the sender and larger signatures. Earlier studies reflect that successful reception of messages over the RDS network is inversely related to the size of the message. Larger signatures result in larger messages and ultimately lowered probability of reception. The computational overhead at the sender for the PCT system can be tolerated since the base station is assumed to be equipped with powerful storage and computing resources. Figure 15 shows the relation between different values of k. As depicted by Figure 15, unsigned messages (no signature or $k = 0$) have a better probability of reception than the signed messages. The messages with larger value for k have lower chances of reception as shown by the curve for $k = 8$ against that of $k = 4$. The reduction in performance is a result of the signatures, which result in large messages being sent over the RDS network. The difference between the signed messages and the unsigned messages is large due to the dependence on successful bootstrapping of the devices.

The results presented in Figure 15 and the rest of the document are for a fixed value of d, where an 8-bit unsigned value carries the index of the values making up the signature. An 8-bit value for the index can represent positions of values for chains up to $d = 256$ values long. It is not expected to have chains more than 256 values in length because that could potentially result in extensive computations at the receivers to verify signatures. To verify a signature, the receiver performs up to $d \cdot k$ hash operations. Large values of d would place a lot of computational burden on the PCT receivers which are expected to have low computational power.

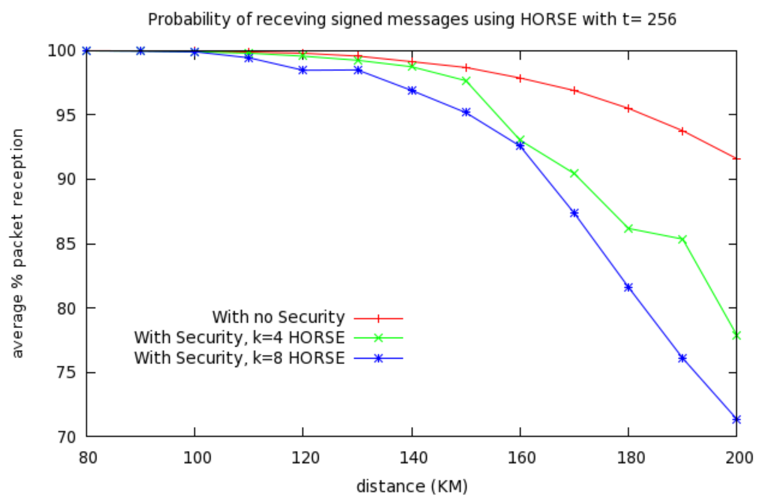


Figure 15: The effects of different signature sizes on reception of messages for HORSE

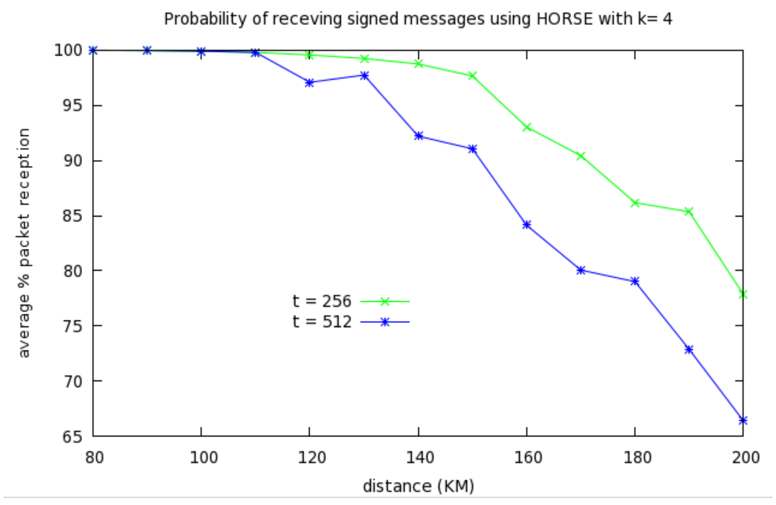


Figure 16: The effects of different public key sizes on reception of messages for HORSE

Successful reception of initial public keys is critical for the verification of signatures. The larger the signatures, the more likely that the initial public key is not received successfully. In this case, the receiver will not be synchronized with the sender and will fail to verify signed messages. Figure 16 shows the effect of varying values of t. The size of the public key grows linear with respect to t. To keep the cases with different values of t equivalent, the value $t \cdot d$ was kept constant in all cases. That is, the amount of storage required by the sender is the same in all cases. The results are as expected with larger values of t performing worse than smaller values. This is shown in Figure 16 with $t = 128$ performing better than $t = 256$ which in turn performed better than $t = 512$.

From the results in Figure 16, it is evident that keeping t as small as possible and keeping d as large as possible improves improve successful initial key reception by reducing the size of the public key. As pointed out earlier, the number of operations at the receiver is equal to $k \cdot d$. For a fixed value of k, increasing d (and reducing k) will result in increased computations at the receivers.

6.3 ECDSA Performance Results

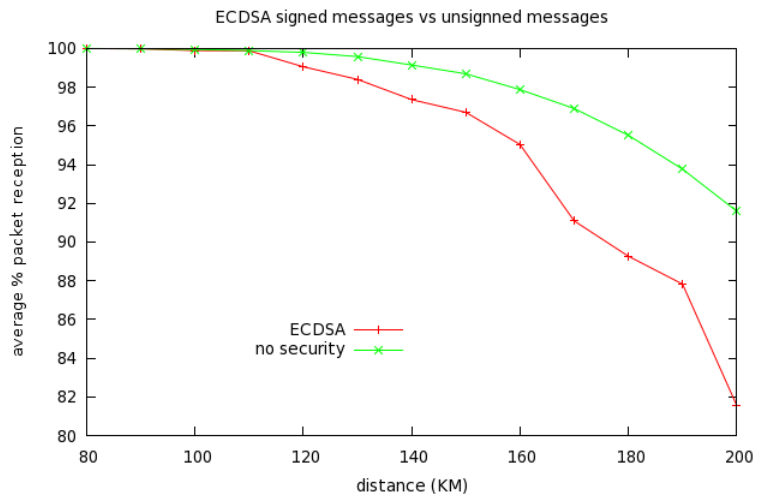


Figure 17: The performance of ECDSA used over the RDS network

The ECDSA protocol produces large signatures compared to BiBa and HORSE. For the recommended 160-bit key (according to [16]) ECDSA has a signature of 40 bytes. This introduces a lot of overhead in the case of PCT messages which are expected to be a few tens of bytes in size. The large messages result in lowered performance as shown by Figure 17. It is worth noting that although the overhead on each message is large for ECDSA, the successful verification of signatures does not depend on receiver bootstrap as is the case with BiBa and HORSE.

6.4 Comparing the Authentication Protocols

Figure 18 puts things in perspective and compares the performance of all the protocols against each other. Figure 18 shows the reception of messages signed by each of the authentication protocols under investigation. From the simulation results as shown in

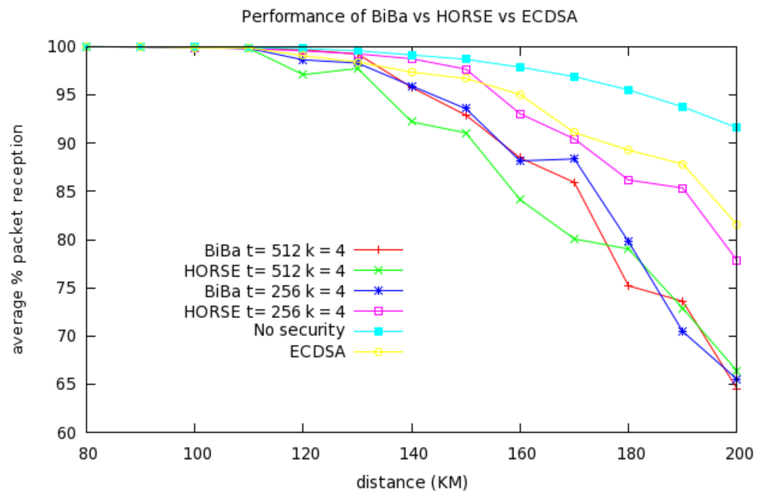


Figure 18: Comparisons between BiBa, HORSE and ECDSA

Figure 18, HORSE performs better than BiBa in terms of successful reception of messages. For the same values of k and t, HORSE has a significant advantage over BiBa as shown in the figure. As an example, at a distance of 14 km, HORSE with $t = 256$ and $k = 4$ gives 5% better message reception than BiBa with the same parameters. The difference in performance increases as the distance increases, reaching 20% at a distance of 18 km. The performance improvement in HORSE comes from efficient use of chain values. In BiBa, the short-term BiBa instance expires when the time set for it elapses even if there are no application messages sent during that period. This results in frequent transmission of the large commitment keys (public keys) which have low chances of successful reception. From the figure, there is no significant difference between HORSE with $t = 256$ and $k = 4$ and ECDSA. It is worth noting that with HORSE the values of t and k can be adjusted to improve performance, albeit with tradeoffs mentioned previously.

ECDSA has large signatures on each message and had overhead of $\frac{40}{30} * 100\% = 133.33\%$ on a 30 byte message (the signature is 40 bytes long). A HORSE instance with parameters $t = 256$, $k = 4$, and $d = 20$ introduces only 41.32% overhead. Such a HORSE instance can be used to sign $d/(1 - e^{-k/t}) = 1290$ messages. Each message is 30 bytes long and the signature on each message is $k(b+a) = 12$ bytes long(where $b = 2$ and $a = 1$; i.e. each value in that make up signature is 2 bytes long and the index variable used for synchronization is 1 byte long). The public key sent to the receivers is $t * 2 = 512$ bytes long, and sent only once. Therefore, the overhead is $\frac{512 + (12)1290}{30(1290)} * 100\% = 41.32\%$. It is important to notice that the overhead of the HORSE protocol will increase with periodic sending of initial public keys to improve bootstrapping probability. The overhead of the BiBa protocol with the same parameters in the best case is equal to $\frac{512 + (8)30\nu d}{30\nu d} * 100\% = 30.08\%$;(with $d = 20$, and $\nu = \lfloor \frac{t\gamma}{k} \rfloor = 25$; typically $\gamma = 0.10$). The best case for BiBa that achieves the above overhead is when the maximum number of messages is signed in each time interval. That is, for the above parameters in each of the $d = 20$ time intervals, $\nu = 25$ messages are signed. The number of signed messages in each interval is expected to be less than the maximum hence the overhead will be larger. The increase in overhead is a result of is inefficient use of keys in BiBa. If the BiBa instances are bootstrapped multiple times to improve successful reception of messages, the overhead is even higher.

The computations that the receivers have to perform when ECDSA is employed for authentication are extensive. In general public key cryptography are computationally extensive because of the arithmetic involved in signature generation and verifications. Public key cryptosystems perform complex operations on relatively large numbers (160 bits for ECDSA). The PCTs are expected to employ low power microcontrollers with limited computational power. Implementations of ECDSA on an 8-bit processor takes 2.78 seconds to verify a signature [19]. On a slightly more powerful 16-bit M16C microcontroller ECDSA was shown to require 630 msec to verify a signature [20]. Comparisons between DSA and HORSE on a PPC 867 MHz platform show that a HORSE implementation using MD5 with $t = 256$, $k = 16$, $d = 2^{10}$ has a key generation time of 0.63 seconds and can verify 2688 signatures in one second [15]. An equivalent DSA has a key generation time of 2.66 seconds and can verify 108 signtures in one second [15] (note that ECDSA is based on DSA). The time required for signature verification in ECDSA is very similar to DSA as shown in [21]. [21] compares software implementations of ECDSA and DSA on a Pentium Pro 200 MHz-based PC to give a perspective on relative performance of ECDSA and DSA. The results show that ECDSA requires 26 ms to verify a signature while DSA require 28.3 ms to verify a signature. HORSE gives a faster signature verification time than DSA (on which ECDSA is based) and a designer faces a tradeoff between the two schemes. The fast verification time of HORSE is ideal for PCT's which are expected to employ microprocessors with modest computational power. The use of HORSE allows for relatively cheap devices at the cost of longer periods to bootstrap the devices. ECDSA does not require bootstrap but requires the devices to perform complex operations and hence take longer times to verify signatures. The time taken to verify a signature could be exploited by an attacker allowing a denial of service attack. If the receivers take a long time to verify a signature, an attacker who floods the network with messages will take up resources at the receivers while they verify the signatures. This could result in missing legitimate messages while the receivers are verifying the bogus messages. Moreover battery powered devices would be unable to go into powersaving modes because they would be kept busy verifying bogus messages, ultimately leading to reduced battery life and increased down times of the receivers.

Security scheme	Overhead	95% service availability (KM)	security level (probability of guessing a valid signature)	computational effort (at receiver)
ECDSA	133.33%	120	2^{-80}	High
HORSE	41.32%	130	2^{-35}	Low
BiBa	30.08%	120	2^{-35}	Low

Table 2: Comparisons of the security schemes for BiBa and HORSE with $t = 512$ and $k = 4$,and ECDSA using SHA-1

Comparisons of the security schemes is presented in Table 2. The table shows the bandwidth overhead introduced by the different security schemes. ECDSA has the most overhead compared to the other schemes. The overhead of HORSE and BiBa as presented in the table is a lower bound and in practice will be higher depending on the number of times public keys are sent to the receivers. The use of signatures makes messages larger which in turn results in increased message loss-rate. Distances from the transmitter at which 95% of the messages are correctly received with the network parameters set as in Table 1 are also presented from simulations. Up to a distance of 90 km all the security schemes achieve high availability of services. Beyond that rane, messages may not be received correctly. The loss of messages is due to signatures and failure to bootstrap receivers for BiBa and HORSE protocols. The security level provided by the schemes in terms of successfully forging a signature by guessing is presented in Table 2. ECDSA gives a high level of security with a cost of added computational complexity at the receivers. BiBa and HORSE offer reduced security and lower computational cost at the receivers. Reversing the one way hash function used in ECDSA, BiBa and HORSE should be computationally infeasible, hence strong hashing algorithms should be employed. An attacker who cannot obtain private key material from the transmitter and tries to forge a signature is limited to guessing. The probability that an attacker can successfully forge an ECDSA signature is 2^{-80} while for HORSE and BiBa it is 2^{-35} [14].

7 Conclusions

Through the study we have shown that security (in the form of message authentication) can be offered over the RDS system. The security provided by the three protocols provides source authentication at a cost of bandwidth efficiency. The construct allows for seamless operation of devices after initialization. The use of a long-term BiBa and HORSE instance allows devices that were rebooted to resynchronize to current short-term instances. The large size of the public keys of the BiBa and HORSE protocol affects the performance, which is dependent on the successful bootstrapping of devices. Although the transmission of the public key for both BiBa and HORSE may take a long time, careful scheduling of such actions for periods with minimal traffic can improve performance. Comparisons shows that the performance of ECDSA is comparable to HORSE. HORSE allows to adjust key sizes and signature sizes to improve performance and reduce overhead. ECDSA on the other hand gives a fixed signature size and introduces significantly higher message overhead compared to BiBa and HORSE. Hybrids of the schemes should be considered for the purposes of making the scheme resilient to attacks and conserve bandwidth.

The protocol employing BiBa and HORSE described here places constraints on the application data rate. The protocol assumes an upper bound of the application data rate. A single BiBa instance can be used to sign a finite number of messages in a single time interval before the security of the protocol falls below targeted levels. In the event that an application exceeds the number allowed by the protocol within a time interval, a decision needs to be made to either buffer such messages until the next period, send the messages unsigned, or sign the message, albeit with reduced security. Buffering some messages may not be ideal for real-time messages, while some critical messages require authentication at the receivers. A more relaxed design is recommended to avoid placing tight constraints on the application. A tradeoff between the public key sizes and the bound on the application message generation rate exist. Smaller public keys means fewer messages can be signed in a single period. Allowing for many messages to be sent in a single period requires that either multiple short-term BiBa instances run in parallel or one short-term BiBa instance with a large number of SEAL chains. The results of both choices effectively increases the size of the commitment keys and ultimately results in increased bootstrapping times and a lowered probability of successful bootstrapping of receivers.

The problem of key distribution still persists with the methods discussed. In the event that a long-term secret key is compromised, there is no feasible way at present to renew it for all the protocols presented. The difficulty is inherent due to the asymmetric nature of the channel. Traditional key agreement and distribution protocols cannot be employed on a one-way communication channel like RDS. Further studies should be conducted to propose manageable ways of system recovery in the event of a long-term chain compromise.

References

[1] E. W. Gunther, "Reference Design for Programmable Communicating Thermostats Compliant with Tite-24." CEC PIER PCT Reference Design, 26 March 2007.

[2] "Security Characteristics of the Title-24 PCT System," 12 April 2007.

[3] M. Daucher, E. Gartner, M. Cortler, W. Keller, and H. Kuhr, "RDS-Radio Data System; A Challenge and a Solution," *Fujitsu Ten Technical Journal*, 30 November 2008.

[4] S. C. Bono, M. Green, A. Stubblefield, A. Juels, A. D. Rubin, and M. Szydlo, "Security Analysis of a Cryptographically Enabled RFID Device," *14th USENIX Security Symposium*, pp. 1–16, August 2005.

[5] A. Juels, "RFID Security and Privacy: A Research Survey," *Selected Areas in Communications, IEEE Journal on*, vol. 24, pp. 381–394, February 2006.

[6] S. W. Ari Juels, *Advances in Cryptology - CRYPTO 2005*, vol. 3621/2005. Springer Berlin / Heidelberg, 2005.

[7] M. FeldHofer, *Cryptographic Hardware and Embedded Systems - CHES 2004*, vol. 3156/2004. Springer Berlin / Heidelberg, 2004.

[8] A. Treytk, N. Roberts, and G. P. Hanke, "Security Architecture for Power-line Metering System," *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pp. 393– 396, 22-24 Septempber 2004.

[9] T. Mander, H. Cheung, A. Hamlyn, and R. Cheung, "Communication Security Architecture for Smart Distribution System Operations," *Electrical Power Conference, 2007. EPC 2007. IEEE Canada*, pp. 411–416, 25-26 October 2007.

[10] P. Gibson, "Keeping scada open and secure," June 2008.

[11] D. K. T. Kunz, "RF Fingerprints for Secure Authentication in Single-Hop WSN," *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing*, pp. 567–573, 12-14 October 2008.

[12] S. J. Engberg, M. B. Harning, and C. D. Jensen, "Zero-knowledge device authentication: Privacy & security enhanced rfid preserving business value and consumer convenience," in *PST*, pp. 89–101, 2004.

[13] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, p. 2002, 2002.

[14] A. Perrig, "The BiBa One-time Signature and Broadcast Authentication Protocol," *Proceedings of the 8th ACM conference on Computer and Communications Security*, pp. 28–37, 5-8 November 2001.

[15] W. Neumann, "HORSE: An Extension of an r-Time Signature Scheme With Fast Signing and Verification," *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 1, pp. 129–134 Vol.1, April 2004.

[16] S. V. Don Johnson, Alfred Menezes, "The Elliptic Curve Digital Signature Algorithm ECDSA." Springer-Verlag, July 21 2001.

[17] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pp. 245–256, April 2008.

[18] R. J. Punoose, V. Pavel, and S. Daniel, "Effecient Simulation of Ricean Fading in a Packet Simulator," *Vehicular Technology Conference*, 2000.

[19] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," *Cryptographic Hardware and Embedded Systems - CHES 2004*, pp. 119–132, 2004.

[20] H. Toshio, J. Nakajima, and M. Matsui, "A Practical Implementation of Elliptic Curve Cryptosystems over GF(p) on a 16-bit Microcomputer," in *PKC '98: Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography*, (London, UK), pp. 182–194, Springer-Verlag, 1998.

[21] J. G. Thomas Wollinger and C. Paar, "Cryptography in Embedded Systems: An Overview," *Proceedings of the Embedded World Exhibition and Conference*, pp. 735–744, February 18-20 2003.